Mars Science Laboratory
Technology Functional Design Document
ROAMS Rover Simulator:
Mobility Dynamics

Revision: Draft 0.6

Date: July 29, 2004

Prepared By:

Garett Sohl

Document Custodian:

Terry Huntsberger

Jet Propulsion Laboratory

4800 Oak Grove Drive

Pasadena, CA 91109-8099

This Page Intentionally Left Blank

MSL Technology Functional Design Document – ROAMS Rover Simulator

Signature Sheet

Approval

Abhi Jain – Technology Provider Date

Terry Huntsberger – Task Manager, Test & Validation: Rover Navigation Date

Revision: Draft 0.6

Date: July 29, 2004

Table of Contents

Revision History

Revision	Date	Description	Author
DRAFT	12/15/02	draft	T. Huntsberger
Initial Release	6/9/04	Initial draft release	Garett Sohl
V0.2	7/13/04	Incorporate changes based on feedback. Added MER EGSE model interface and removed ROAMS controller and profile generator models.	Garett Sohl
V0.3	7/20/04	Further edits based on feedback. Changed penetration distance computation description (section 5.1.9.1) to conform to analytic model instead of SWIFT++ based model. This analytic model is now the default model used by ROAMS.	Garett Sohl
V0.4	7/23/04	Removed description of IMU model that included noise and bias terms – testing will use the “truth” IMU model.	Garett Sohl
V0.5	7/26/04	Removed specific references to the MER EGSE model and replaced with generic commander/controller. Included equations for models with non-trivial algorithms as replacements for web links. Added “Scope” section after introduction to outline responsibilities of this document and the ICD. Removed ephemeris model description since it is not part of mobility dynamics.	Garett Sohl
V0.6	7/29/04	Added working requirements section. Removed some references to the specific interface layer used for testing since that is covered in ICD. Added subscripts for some equations. Changed sign in joint stop spring equation to match ROAMS convention.	Garett Sohl

1 Introduction

This document contains the Functional Design Description (FDD) for the mobility dynamics portion of the ROAMS rover simulation software. The ROAMS software includes rigid body kinematics and dynamics, motor/encoder models, an IMU model, a model of the rover environment and algorithms that describe how the rover interacts with the environment. The primary goal of ROAMS is the modeling of rover surface operations.

2 Scope

This document provides a functional description of ROAMS mobility dynamics. While ROAMS provides several functional layers, this document describes only those algorithms and models used to simulate the mobility dynamics of a rover on a defined surface. The high-level navigation, obstacle avoidance, camera and odometry models used by ROAMS are not described in this document.

The Interface Control Document (ICD) (see Table 1 – Related Documents) defines the interface to the ROAMS mobility dynamics system and lists the set of user-configurable vehicle and model parameters.

3 Functional Requirements

This FDD is based on the following working requirements:

- Vehicle Dynamics: ROAMS will simulate the multi-body, articulated dynamics of the rover. This model includes mechanical joint stops and the differential constraint for the rocker arms.
- Actuators: ROAMS will provide a model of the DC motors used for driving and steering the rover. Input to the motor model will be motor voltage.
- Sensors: ROAMS will provide truth-based, zero-noise sensor models for encoders, potentiometers and IMU.
- Terrain Modeling: ROAMS will model the terrain used for driving. A digital elevation map will be used to model terrain height.
- Wheel-soil Interaction Modeling: ROAMS will model wheel-soil interaction using a compliance contact model. Soil parameters will include soil density, cohesion and internal friction angle. The contact model allows rolling and sliding types of contacts on a per-wheel basis.

MSL project requirements for ROAMS are currently under development (MSL Technology Requirements Document in Table 1 – Related Documents). Future ROAMS requirements will be derived from these project requirements.

4 Related Documents

The documents related to ROAMS are given in Table 1

Table 1 – Related Documents

Documents
ROAMS: Rover Analysis Modeling and Simulation Software, J. Yen and A. Jain. I-SAIRAS ‘99. Noordwijk, Netherlands. June 1999
ROAMS: Planetary Surface Rover Simulation Environment, A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, R. Steele. I-SAIRAS ’03. Nara, Japan. May 2003
Recent Developments in the ROAMS Planetary Rover Simulation Environment. A. Jain, J. Balaram, J. Cameron, J. Guineau, C. Lim, M. Pomerantz, G. Sohl. IEEE 2004 Aerospace Conference. Big Sky, Montana. March 2004.
DARTS/Dshell web page: http://dartslab.jpl.nasa.gov
MSL Technology Requirements Document: ROAMS Mobility Dynamics

5 Hardware Interfaces and functionality

ROAMS does not interface to rover H/W. ROAMS provides a simulation of rover hardware components.

6 Basic Software Functionality

6.1 ROAMS Functional Component Overview

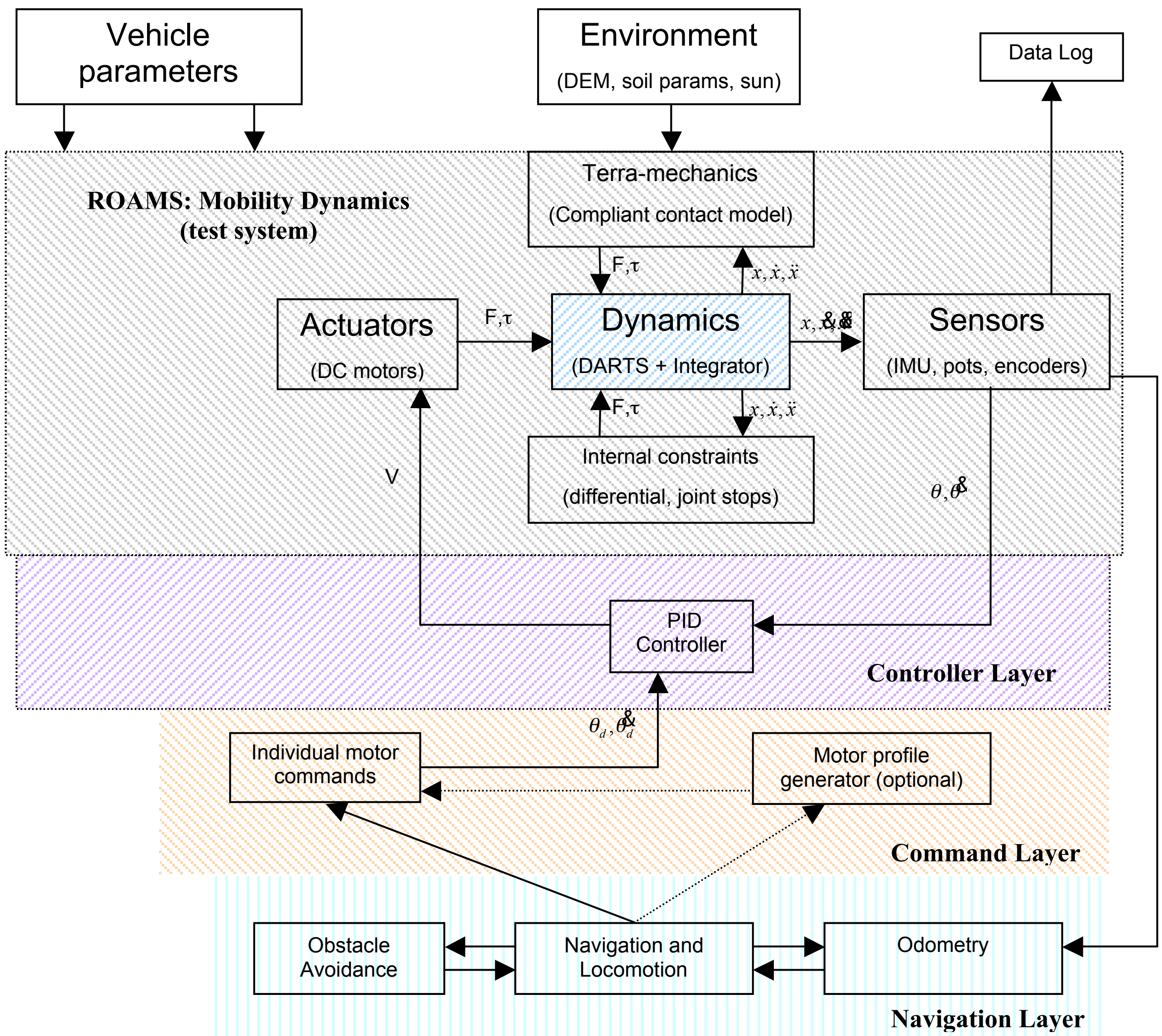

Figure 1 – ROAMS SW Overview

This section describes the nominal software functionality required for ROAMS. Figure 1 provides an overview of the various components and functional layers of ROAMS. The components of the mobility system include the vehicle dynamics, sensor, actuator and environmental models. ROAMS can provide a user interface to the controller, command or navigation layers shown in Figure 1.

6.1.1 Dynamics Algorithms (DARTS) and Integration (CVODE)

In order to simulate the rover dynamics, ROAMS uses DARTS (Dynamics And Real-Time Simulation) as shown in Figure 1. DARTS provides an efficient dynamics formulation for muti-body, articulated systems. The rover kinematic joint locations, mass, and inertia properties are input parameters to DARTS. Given the forces and torques (F,τ) acting on the rover, DARTS computes the accelerations of each articulated body. These accelerations are then integrated forward in time to propagate the rover state ($x, \dot{x}, \ddot{x}$). ROAMS typically uses a variable step, stiff integrator called CVODE. CVODE was developed at Lawrence Livermore National Labs as part of SUNDIALS (Suite of Nonlinear and Differential Algebraic equation Solvers) and is based on previous Fortran integrators VODE and VODPK. ROAMS normally operates using an integration step size of 0.05 seconds, but it should be noted that CVODE is an adaptive integrator and may take many internal steps before advancing the requested 0.05 seconds of simulation time.

6.1.2 Generic Model Interface (Dshell)

The Dshell (DARTS shell) package provides a generic interface between the underlying physics in DARTS and a variety of models. ROAMS implements several models in the Dshell environment as show in the grayed region of Figure 1. The Dshell models used in ROAMS include internal constraint models, actuator/sensor device models, controller models, and a compliance model of the wheel-soil contact.

6.1.3 Internal Constraint Models

The rocker differential employed by most rovers forms an internal closed-chain articulation and constrains the right and left rocker arms to deflect an equal amount in opposite directions. ROAMS uses a Dshell-based, compliant constraint model for the rocker differential. This constraint model applies an internal restoring force using a stiff spring to prevent violation of the equal and opposite rocker constraint. ROAMS uses a linear compliance model for the differential joint:

$$\tau_R = k_R e + c_R \dot{e},$$

where τ_R is the differential rocker constraint torque, k_R is the spring constant, c_R is the damping constant, e is the sum of the left and right side rocker angles and $\dot{e}$ is the time derivative of e.

ROAMS employs a similar technique to handle joint limits on rocker/bogey arms. A compliant opposing torque is applied when a joint proceeds past a joint stop. In order to prevent significant violations of either joint stop or the rocker differential constraint, the compliance constraints employ a very stiff spring model. ROAMS uses a non-linear (Hunt-Crossley) spring model for the joint stop system:

$$\tau_J = -k_J |\delta_J|^n sign(\delta_J) - \frac{3}{2}\alpha_J \dot{\delta}_J |\delta_J|^n,$$

where the subscript J denotes joint stop, $\delta_J > 0$ is the rotation past the upper joint limit, $\delta_J < 0$ is the rotation past the lower joint limit, $sign(\delta_J)$ is +1 or -1 (depending on the sign of δ_J), τ is the constraint torque, k is the spring constant, α is the damping coefficient and n is the non-linear exponential.

Note that while a larger k value for either of these constraints will reduce the constraint violation, choosing k too large can cause numerical integration problems. As k increases, the performance of the numerical integrator will decrease as it is forced to take smaller internal steps to integrate the stiff system.

6.1.4 Actuator Models

ROAMS uses individual DC motor models to provide actuation force for each steering, wheel and arm joint on the rover. Parameters for the motor model are the motor armature constant, stall torque, resistance and mechanical damping. Input to the model is the motor voltage. The DC motor models apply torques to the rover dynamics as show in Figure 1. Motor current dynamics are assumed instantaneous; we assume the motor inductance is equal to zero. Algorithmically, the motor model first computes the current draw using Kirchoff's law (assuming inductance is zero):

$$Ri = V_{in} - K_t\dot{\theta} \Rightarrow i = \frac{1}{R}\left(V_{in} - K_t\dot{\theta}\right),$$

where K_t is the armature constant (in SI units), $\dot{\theta}$ is the rotational velocity of the shaft, V_{in} is the input voltage, and i is the motor current draw. Based on the current draw, we can compute the motor torque as:

$$\tau = K_t i - b\dot{\theta},$$

where b is the mechanical damping constant and τ is the motor torque.

6.1.5 Sensor Models (IMU, potentiometers, encoders)

ROAMS uses encoder and potentiometer models that provide joint angle and rate information for the rover wheel, steering, arm and rocker/bogey joints. Normally, these models output truth data from the simulation. However, a reset signal allows the user to initialize the current output angle to be zero. Parameters also allow overriding the positive angle direction of any encoder/potentiometer. The current encoder and potentiometer models do not include any noise.

ROAMS provides a “truth” IMU model that outputs truth (zero noise, zero bias) values for linear and angular rates and accelerations. While ROAMS also includes more accurate IMU models, the truth-based model will be used for validation tests.

6.1.6 Environmental Models

In addition to vehicle modeling, ROAMS also models the rover environment. As a surface vehicle, the rover interacts with the environment primarily through the terrain. A Digital Elevation Map (DEM) represents the terrain (including rocks) in ROAMS. A DEM represents a single height value (z) for any given location (x,y). ROAMS uses the SimScape software package to access DEM-based terrain data. SimScape is currently under development at JPL. SimScape allows ROAMS to select from a set of pre-defined DEM-based terrains and provides an API to query for height and slope values at any point on the terrain.

ROAMS parameterizes soil mechanics properties using density, internal friction angle and soil cohesion. The user may explicitly set these parameters or select parameter sets that match certain soil types (clay, loose sand, mixed, etc.). ROAMS will support spatial variations of these soil parameters via three separate PGM overlay files. These overlay files will define the variation of soil parameters over the surface of the terrain.

ROAMS allows the user to define the planetary location of the DEM. A terrain DEM can be placed at a given latitude/longitude on the surface of either Mars or Earth. ROAMS will automatically set the gravitational acceleration based on the planet choice.

6.1.7 Wheel-soil Contact Model

The primary goal of the wheel-soil contact model is to compute the forces and torques exerted by the soil on the rover vehicle. Given these forces, DARTS can compute the resulting accelerations for the rover using the multi-body dynamics model. For a 6-wheel rover, there are 36 unknowns: 3 forces and 3 torques for each of six wheels. However, a static force/moment balance for the rover-terrain system yields only 6 equations. Constraint equations written about each bogey and about the rocker differential yield an additional 3 equations. The problem is therefore statically indeterminate (9 equations in 36 unknowns) and there is a large solution space of contact forces/torques that satisfies the equations.

ROAMS makes some assumptions to reduce the number of unknowns. First, ROAMS assumes single point contact for each wheel. This assumption eliminates half of the unknown variables since torque can not be exerted at a point contact. We are still left with 3 unknown forces at each wheel contact point. To solve for these forces, ROAMS uses a compliance contact system. We use two separate and independent compliance systems at each wheel contact to compute the unknown forces. One compliance system is used to compute the normal force and the second system is a two degree of freedom system used to compute the tangent force. The spring and damping coefficients for these compliance systems are parameters of the compliant contact model. The compliance model uses Terzaghi's terra-mechanics equations to compute the maximum available traction force at each contact and to estimate the wheel sinkage into the terrain. The following sections describe how the wheel-soil contact model computes normal and tangent forces.

6.1.7.1 Normal Force Computation

The magnitude of the normal force is the foundation of most contact models. The magnitude of the normal force is used to compute the available traction force in the tangent plane. The normal force is defined as the component of the force in the direction normal to the plane of the terrain at the contact point. The terrain local normal is computed by averaging the normal direction for a small patch underneath each wheel. The size of this patch is user configurable. The terrain is then assumed to be locally planar and perpendicular to the computed normal direction. New local normal directions are computed every simulation step. ROAMS does not currently model the effect of terrain deformation on the normal direction.

In order to compute the force in the normal direction, ROAMS uses a single degree of freedom, Hunt-Crossley compliance system at each wheel contact point. The force, *F*, for the non-linear Hunt-Crossley model is given as:

$$F_{N,i} = k_{N,i}\delta_{N,i}^{n} + \frac{3}{2}\alpha_{N,i}\dot{\delta}_{N,i}\delta_{N,i}^{n},$$

where the subscript *N,i* denotes the normal direction for wheel *i*, *k* is the spring constant, δ is the deflection, α is the damping coefficient and *n* is the deflection exponent.

The deflection of the compliance system is based on the penetration of the wheel into the terrain in the normal direction. The location of the contact point is defined as the point on the wheel surface that penetrates the farthest into the terrain. As a wheel begins to penetrate the terrain, the compliance system applies an opposing force at the contact point based on the penetration distance. ROAMS uses a very stiff spring constant to prevent unrealistically large penetration distances. This can, at times, lead to numerical integration problems. The user can set the spring and damping parameters for each wheel and some tuning of these parameters (based on total rover mass) can greatly improve integration performance and stability.

Efficient computation of the penetration distance and the contact point (point of maximum penetration) for two geometric shapes (wheel and terrain) is a complex problem. There are few, if any, publicly available software packages that directly compute these quantities for general, convex polytopes. ROAMS uses an analytic penetration distance model to compute penetration distance and the contact point. This analytic model approximates the shape of the wheel using a line segment and a radius. The line segment is the rotation axis of the wheel and the length of the segment is the width of the wheel. The distance between this line segment and the locally planar terrain is computed analytically and differenced with the wheel radius to obtain the penetration distance. The contact point on the wheel is computed based on the end of the line segment closest to the terrain and the terrain normal direction. Note that this analytic model doesn't exactly match the shape characteristics of a cylindrical wheel since the cylinder ends are hemispherical instead of flat.

6.1.7.2 Tangent Force Computation

While the normal direction forces provide "support" for the vehicle, the tangent plane forces provide the motive force for a wheeled vehicle. A two dimensional compliance system is used in the tangent plane. ROAMS typically uses a linear spring system for the tangent force, *F*:

$$F_{T,i} = k_{T,i}\delta_{T,i} + \alpha_{T,i}\dot{\delta}_{T,i},$$

where the *T,i* subscript denotes the tangent direction for wheel *i*, *k* is the spring constant, α is the damping coefficient and δ is the deflection vector. The tangent system deflection vector is based on the relative velocity of the contact point when the wheel is rolling. As the contact point on the wheel begins to move relative to the terrain, the tangent plane compliance system is deflected in the opposite direction of this motion:

$$\dot{\delta}_{T,i} = -V_{T,i} \,,$$

where the *T,i* subscript denotes the tangent direction for wheel i, δ is the deflection of the 2-D tangent compliance system and *V* is the velocity of the contact point relative to the surface.

Based on the magnitude of the normal force, there is a limit on the available traction force in the tangent plane. For simple Coulomb friction, the limit is given as:

$$\left\| F_{T,i} \right\| \le \mu_i F_{N,i} \,,$$

where μ is the friction coefficient. On soil, the traction limit is typically based on soil internal friction angle and cohesion:

$$\left\| F_{T,i} \right\| \le c_i + F_{N,i} \tan \psi_i = F_{\max,i} \,,$$

where *c* is the soil cohesion, ψ is the internal friction angle and *i* denotes the wheel number. The traction limit defines the transition point between rolling (tangent force less than traction limit) and sliding contact (tangent force equal to traction limit) for each wheel. Once this limit is reached, the wheel begins to slip. The tangent plane compliance system is constructed to converge to the correct tangent force for rolling contact (where the contact point has zero velocity relative to the terrain) and to transition to sliding contact (where the contact point has non-zero velocity relative to the terrain) when the maximum traction force is reached. The following procedure is used:

1. Assume contact is rolling and compute F_T and F_N based on the current deflection in the normal and tangent directions.

$$F_{N,i} = k_{N,i}\delta_{N,i}^{n} + \frac{3}{2}\alpha_{N,i}\dot{\delta}_{N,i}\delta_{N,i}^{n}$$

$$F_{T,i} = k_{T,i}\delta_{T,i} + \alpha_{T,i}\dot{\delta}_{T,i}$$

2. If $\left\| F_{T,i} \right\| \ge F_{\max,i}$, contact is **sliding**. Set $F_{T,i} = F_{\max,i}$ and invert the compliance equation of the tangent system to compute $\dot{\delta}_{T,i}$:

$$\dot{\delta}_{T,i} = \frac{F_{T,i} - k_{T,i}\delta_{T,i}}{\alpha_{T,i}}$$

3. If $\left\| F_{T,i} \right\| < F_{\max,i}$, contact is **rolling**. Compute $\dot{\delta}_T$ based on relative velocity of contact point:

$$\dot{\delta}_{T,i} = -V_{T,i}$$

6.1.7.3 Sinkage Estimation

The contact model also computes a sinkage estimate for each wheel-soil contact and stores this as a user readable model state. This may seem paradoxical as we have already described how the normal force is computed based on the penetration of the wheel into the terrain. It is important to understand, however, that the wheel-soil penetration used to compute the normal force is a function of the spring and damper coefficients for the compliance system. Based on the computed normal force, we can compute an estimate of the actual wheel sinkage based on terra-mechanics equations. ROAMS uses Terzaghi's empirically derived sinkage equations. Terzaghi computes the bearing capacity of soil as:

$$q = 1.3cN_c + \gamma D_f N_q + 0.4\gamma B N_\gamma \,,$$

where q is the bearing capacity of the soil (kg/m^2), c is the soil cohesion, γ is the soil density, B is the width of the wheel and D_f is the sinkage of the wheel. N_c, N_q and N_γ are the cohesion, surcharge and friction angle coefficients respectively; these coefficients are derived empirically by Terzaghi based on the internal friction angle of the soil. At equilibrium, we have:

$$F_{N,i} = Aq\ ,$$

where A is the area of the wheel-soil contact patch and q is the bearing capacity of the soil. Since the contact patch area depends implicitly on the sinkage, ROAMS uses a single step Newton-Raphson solver to estimate the sinkage based on the current normal force, soil parameters and wheel geometry.

The sinkage value will be used to estimate rolling resistance for each wheel as it deforms the local terrain. The algorithms used to estimate rolling resistance are currently under development by the ROAMS team.

6.2 Command Interface

The main command input to ROAMS for testing is through the controller layer as shown in Figure 1.

6.2.1 Error Sources for Motor Profile Tracking

The motor controller performance depends on:

- PID controller gains: if controller gains are too small, tracking performance will be poor.
- Motor torque limits: the motor model may stall and fail to exert enough torque when climbing steep slopes or when large accelerations are commanded.
- Wheel-soil contact point: for very rough terrain, the wheel-soil contact point may change rapidly. Since the contact force acts as a disturbance for the motor controller, these rapid changes can cause poor tracking performance.

6.3 Engineering Data Products

ROAMS internally manages a large amount of data consisting of the full rover state as well as the current parameters, inputs and outputs of all models. ROAMS will output the current wheel and steering motor states to the controller. ROAMS will record simulation data to an internal buffer at each integration step. The simulation data is logged to a file when the internal data buffer becomes full. The integration step size, which controls the data-logging rate, is configured at run time. ROAMS allows the user to activate and deactivate data logging at run time.

7 Fault Modeling

ROAMS does not currently support fault injection into the simulation models. This feature may be added at a later time if required. ROAMS does monitor and log threshold conditions for joint stops and motor torques.

8 Operational Scope

At outlined in this document, ROAMS consists of a wide array of models. These models embody tradeoffs between fidelity and a variety of factors including speed of execution on current computer hardware and model complexity. Each ROAMS model is designed to work in a certain operational regime. This section provides the user an overview of the current operational scope of the models and algorithms used in ROAMS.

- Simulation clock advancement: ROAMS can either control the advancement of the simulation clock internally or interface to an external program that interactively (e.g. via IPC) advances the simulation clock for small increments of time. Speed of simulation advancement in comparison to real-time depends on many factors: integration step size, spring stiffness, CPU speed, etc.

- Integration step size: ROAMS uses the CVODE stiff integrator with a default integration step size of 0.05 seconds. The user can modify this step size using the override file.
  - An overly large step size can cause the numerical integrator to become unstable.
  - Integration step size determines how often a new terrain normal is computed for each wheel-soil contact point. Choosing a step size that is too large can cause terrain height and slope changes to be smoothed over by ROAMS.
  - A smaller step size will increase computational load and slow down the advancement of the simulation clock compared to real time.
- Vehicle parameters: The vehicle parameters are not all independent. When overriding default parameters, the user must provide ROAMS with a consistent parameter set.
  - For example, increasing the mass of the rover chassis also requires modifying the spring constant for the rocker differential constraint to prevent overly large violation of the differential constraint.